Problem A. Divide and Conquer

Time limit 1000 ms
Mem limit 262144 kB

An array b is *good* if the sum of elements of b is even.

You are given an array a consisting of n positive integers. In one operation, you can select an index i and change $a_i := \lfloor \frac{a_i}{2} \rfloor$.[†]

Find the minimum number of operations (possibly 0) needed to make a good. It can be proven that it is **always** possible to make a good.

[†] $\lfloor x \rfloor$ denotes the floor function — the largest integer less than or equal to x. For example, $\lfloor 2.7 \rfloor = 2$, $\lfloor \pi \rfloor = 3$ and $\lfloor 5 \rfloor = 5$.

Input

Each test contains multiple test cases. The first line contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \leq n \leq 50$) — the length of the array a.

The second line of each test case contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^6$) — representing the array a.

Do note that the sum of n over all test cases is not bounded.

Output

For each test case, output the minimum number of operations needed to make a good.

Sample 1

Input	Output
4 4 1 1 1 1 2 7 4 3 1 2 4 1 15	0 2 1 4

Note

In the first test case, array a is already *good*.

In the second test case, we can perform on index 2 twice. After the first operation, array a becomes $[7, 2]$. After performing on index 2 again, a becomes $[7, 1]$, which is *good*. It can be proved that it is not possible to make a *good* in less number of operations.

In the third test case, a becomes $[0, 2, 4]$ if we perform the operation on index 1 once. As $[0, 2, 4]$ is *good*, answer is 1.

In the fourth test case, we need to perform the operation on index 1 four times. After all operations, a becomes $[0]$. It can be proved that it is not possible to make a *good* in less number of operations.